Evolutionary Learning Strategy using Bug-Based Search

Hitoshi IBA[1] Tetsuya HIGUCHI[2] Hugo de GARIS[3] Taisuke SATO[1]
1)Machine Inference Section,
2)Computational Models Section,
Electrotechnical Laboratory,
1-1-4 Umezono, Tsukuba-city,
Ibaralci, 305, Japan
{iba,higuchi,sato}@etl.go.jp

3)Brain Builder Group,
ATR Human Information Processing
Research Laboratories,
2-2 Hikari-dai, Seiko-cho, Soraku-gun,
Kyoto, 619-02, Japan.
degaris@hip.atr.co.jp

Abstract

We introduce a new approach to GA (Genetic Algorithms) based problem solving. Earlier GAs did not contain local search (i.e. hill climbing) mechanisms, which led to optimization difficulties, especially in higher dimensions. To overcome such difficulties, we introduce a "bug-based" search strategy, and implement a system called BUGS2. The ideas behind this new approach are derived from biologically realistic bug behaviors. These ideas were confirmed empirically by applying them to some optimization and computer vision problems.

1 Introduction

This paper introduces a new GA (Genetic Algorithms) based approach to evolutionary learning. The purpose of our research is to apply artificial life systems to practical problem solving or to establish problem solving from nature.

Traditional GAs optimize functions by adaptive combination (crossover) and mutation of coded solutions to problems (i.e. points in the problem's search space) [Goldberg89]. These genetic search mechanisms often suffer from optimization difficulties cased by premature convergence [Schaffer91] or hamming cliff [Caruana88]. Although some recent works intend to realize local search operators for GAs [Ackley87][Muhlenbein89], it is very difficult to switch gradually from global search to local search by adaptive methods. Moreover, as shown in [Rechenberg86], for search in higher dimensions typical in computer vision applications (e.g. see the 11 dimensional search space of Fig.6) there is little hope in using random search or simple recombination methods. However, even for a higher dimensional space, in general the essential search dimensions are relatively few. Thus the best way is to climb up the gradient hill in these essential dimensions.

To solve these difficulties, we present an evolutionary learning system, called BUGS2. In our earlier paper, we described the implementation of BUGS, a bug-based search system using GAs [Iba92a]. The basic idea of BUGS is that the GA chromosomes represent directional control codes rather than positional vectors and that selection criteria is based on cumulative fitness. These are derived from evolutionary learning models of predatory behaviors [Iba92c]. BUGS2 extends BUGS by incorporating more biologically realistic bug behaviors, such as sexual/asexual reproductions, variable size of population, resource sharing and resource competition. In the extended strategy an analogy is made between the value (at a given point) of a function to be maximized, and the density of bacteria at that point. As a result, the usual GA adaptive search method is integrated more naturally with hill climbing search. These ideas are empirically confirmed by applying them to optimization and computer vision problems.

2 Evolution of Predatory Behaviors using Genetic Search

This section introduces the fundamental idea that our BUGS2 program is based on. We experimented with the evolution of bugs which possess "predatory behaviors", i.e. the evolution of bugs which learn to hunt bacteria. The original motivation for these experiments was derived from [Dewdney89]. Bugs learn to move to those regions in the search space where the bacterial concentration is highest. Since the bug concentration is set up to be proportional to the local value of the function to be maximized in the search space, the "stabilized" bug concentrations are proportional to these search space values. Hence the bugs learn (GA style) to be hill climbers.

2.1 Bugs hunt bacteria

Fig.1(a) illustrates the world in which our bugs (large dots) live (a 512x512 cellular grid). They feed on bacteria (small dots) which are continually being deposited. Normal bacterial deposition rate is roughly 0.5 bacterium per (GA) generation over the whole grid. Each bug has its internal energy source. The maximum energy supply of a bug is set at 1500 units. When a bug's energy supply is exhausted, the bug dies and disappears. Each bacterium eaten provides a bug with 40 units of energy, which is enough to make 40 moves, where a move is defined to be one of six possible directional displacements of the bug as shown in Fig.2.

A bug's motion is determined by coded instructions on its gene code.vSix directions a bug can move are labeled F, R, HR, RV, HL and L for Forward, Right, Hard Right, Reverse, Hard Left, and Left. The GA chromosome format for these bugs is an integer valued 6-vector expressed

in their order (F,R,HR,RV,HL,L), e.g. (2,1,1,1,3,2) (as shown in the window on Fig. 1(a)). When a bug is to make a move, it will move in direction d_i (e.g. d_3 = *HR*) with a probability p(d»), which is determined by the formula $p(d_i) = e^{ai} / \sum_{j=1}^{6} e^{aj}$, where a_i is the i-the component value of the 6-vector (e.g. a_5 = 3 above). Once a move is made, a new directional orientation needs to be made. Fig.2 shows the new F_{next} directions, e.g. if the move is R, new forward direction will be to the right (i.e. east). For instance, a bug with a gene code of (1,9,1,1,1,1) turns frequently in direction R that it is highly likely to move in a circle.

After 800 moves (i.e. when it attains an "age" of 800), the bug is said to be "mature" and is ready to reproduce if it is "strong" (i.e. its energy is greater than some threshold value of 1000 energy units). There are two types of reproduction, asexual and sexual. With asexual reproduction, a strong mature bug disappears and is replaced by two new bugs (in the same cell on the grid). Each daughter bug has half of the energy of its parent. The genes of each daughter bugs are mutated as follows. One of the components of the directional 6-vector is chosen with uniform probability. The value of the direction is replaced by a new value chosen with uniform probability (over the integer range of e.g. [0,10]). Sexual reproduction occurs when two strong mature bugs "meet" (i.e. they move within a threshold distance from each other called the "reproductive radius"). The distance between two parents is defined as the Euclidean distance between the two parents. The reproductive radius is set at 10.0. The two parents continue to live and are joined by the two daughter bugs, where one daughter is placed in the grid position of one of the parents and similarly for their other daughter. Each parent loses half its energy in the sexual reproductive process. As a result, two children are born, whose energies are half of the average of parents' energies. The children's genes are obtained by applying mutation and uniform crossover operators to the parents' genes. Thus these reproductions are constrained with probabilities.

Fig.1(b) shows the results of the first simple experiment. The simulation began with 10 bugs with random genetic structures. Most of the bugs jittered from side to side unpredictably and are called "jitterbugs". They are likely to starve to death because they eat up most of the food in their immediate vicinity and are unable to explore more widely. In time "cruiser" bugs evolve, which move forward most of the time and turn left or right occasionally. (Note that if a bug hits an edge of the grid, it stays there until an appropriate move displaces it away from that grid edge.) These "cruiser" bugs succeed in finding food and thus dominate the entire population. A typical chromosome for a "cruiser" bug is shown in the sub-window of Fig.l(b); i.e. (9,6,0,2,4,1). The remarkable features of this chromosome (6-vector) are as follows.

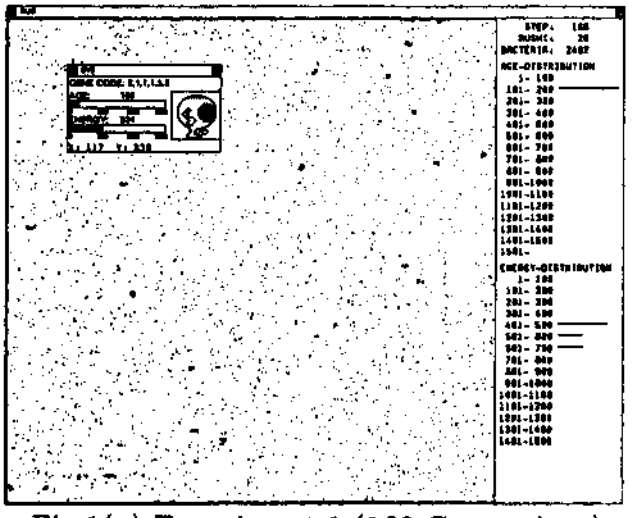


Fig.1(a) Experiment 1 (166 Generations)

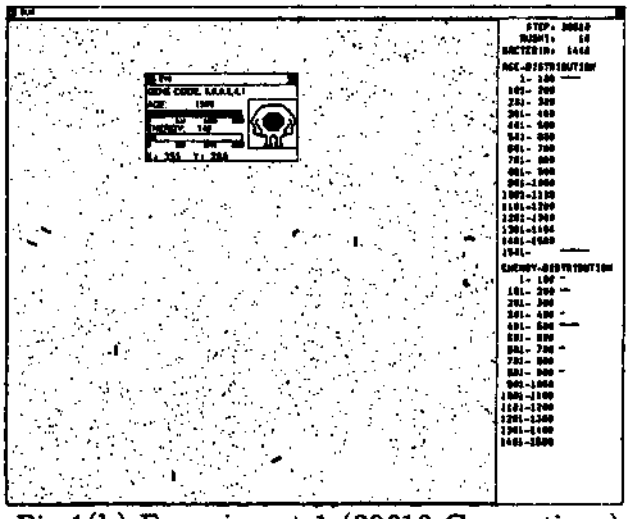


Fig.1(b) Experiment 1 (39618 Generations)

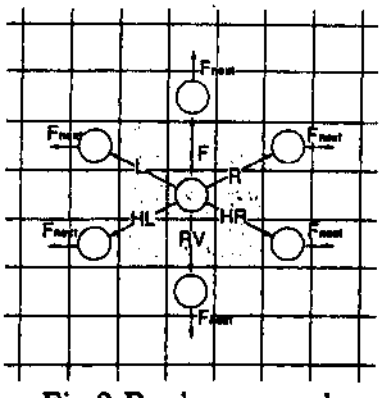


Fig.2 Bug's gene code

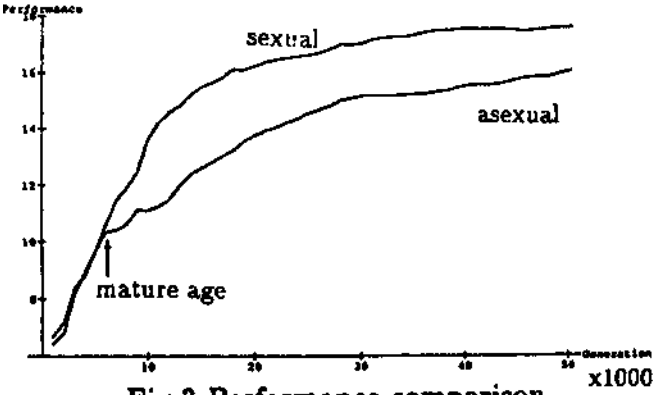


Fig.3 Performance comparison

1. The forward gene (F) is large (9).
2. The reverse gene (RV) is small (2).
3. One of the Right (R), Left (L), Hard Right (HR) and Hard Left (HL) is of middle size (6).

The second feature is important because bugs with large "reverse" (RV) gene values create "twirlers" which make too many turns in one direction. Such unfortunate creatures usually die. The third feature is also essential, because intelligent bugs have to avoid loitering around wall edges.

2.2 Effectiveness of sexual reproduction

Dewdney's original paper used only mutation operators, i.e. asexual reproduction. We introduce sexual reproduction to increase the effective evolution of our bugs. To show that the speed of evolution is higher with sexual reproduction, we conducted some experiments. In the first experiment, we statistically compared the performance rates, where the performance of bugs at the generation *t* is defined as follows:-

$$Performance(t) = \sum_{i=0}^{9} Perf(t-i), \quad where \qquad (1)$$

$$Perf(k) = \frac{\#Eaten(k)}{\#Bac(k) \times \#Bug(k)} \qquad (2)$$

$$\#Bug(k) = (no.\ of\ bugs\ at\ the\ k-th\ generation) \qquad (3)$$

$$\#Bac(k) = (no.\ of\ bacteria\ at\ the\ k-th\ generation) \qquad (4)$$

$$\#Eaten(k) = (no.\ of\ bacteria\ eaten\ by\ bugs\ at\ k-th\ gen.) \qquad (5)$$

This performance indicates how many bacteria are eaten by bugs as a whole for the previous 10 generations. As can be seen in Fig.3 , sexual reproduction after the mature age (800) performs better than asexual reproduction, which is tested statistically (see [Iba92d] for more details).

In a second experiment, the bacteria in the lower left-hand corner (called the Garden of Eden, a square of 75x75 cells) are replenished at a much higher rate than normal (Fig.4(a)); normal bacterial deposition rate is roughly 0.5 bacterium per (GA) generation over the whole grid. In the Garden of Eden, this rate is 0.5 over the 75x75 area, i.e. a rate roughly $\frac{512 \times 512}{75 \times 75} \approx 47$ times as great. As the (GA) generations proceeded, the cruisers evolved as before. But within the Garden of Eden, the jitterbugs were more rewarded for their jittering around small areas. Thus two kinds of "species" evolved (i.e. cruisers and twirlers) (Fig.4(c)). Note how typical gene codes of these two species differed from each other. In this second experiment, we compared three different strategies (asexual reproduction, sexual reproduction, and sexual reproduction within reproductive radius) in four different situations. The aim is to evolve a mix of bugs, namely "cruisers" and "twirlers". We test two initial conditions; a) randomized initial bugs and b) "cruisers" already evolved. In addition, the influence of an empty area in which no bacteria exist is investigated. Obviously this empty-area condition makes the problem easier. The results of these experiments are as follows:-

Task: initial	Task: empty area	Asexual: mutation	Sexual: crossover mutation	Sexual: Prox. crossover mutation
Random	○	∘	○	⊙
Cruisers	○	∘	○	⊙
Cruisers	×	△	△	○
Random	×	△	△	○

△ difficult ∘ possible ○ fast ⊙ faster

Table 1 Experimental results

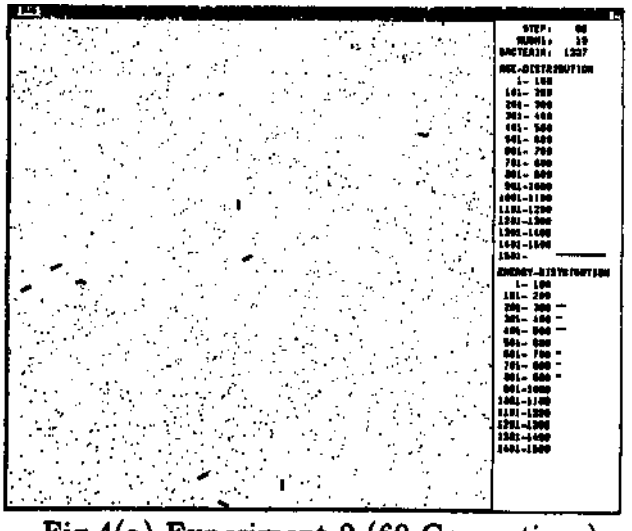


Fig.4(a) Experiment 2 (69 Generations)

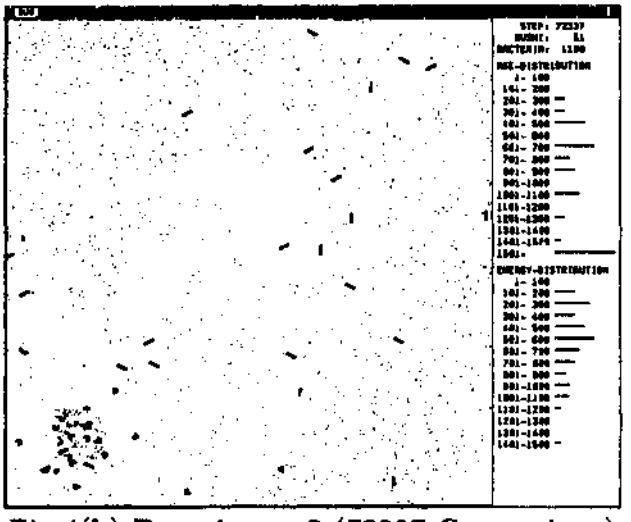


Fig.4(b) Experiment 2 (72337 Generations)

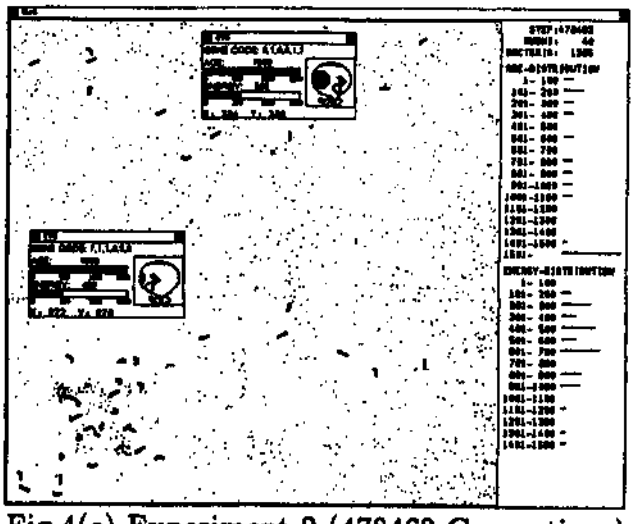


Fig.4(c) Experiment 2 (478462 Generations)

As shown in the table, the sexual reproduction with a reproductive radius is superior to the other two strategies and the improvement is significant for more difficult tasks such as no empty-area conditions.

Thus we have confirmed that the crossover is useful for the evolution of predatory behavior. The described method contrasts with traditional GAs in that our approach uses search directions rather than positions and that selection is based on energy. This idea leads to a bug-based GA search (called BUGS2) whose implementation is described in the next section.

3 BUGS2 : A Bug-Based Search Strategy

In this section, we apply the ideas and techniques introduced in the previous section to optimization problems. The formalism introduced is general enough to be applicable to a 11-dimensional problem in section 4. As mentioned in section 1, the main idea of this paper is to use "bugs" as function optimizers, where the bacterial concentration in a region is proportional to the function value at that region. The function to be maximized is defined as :-

$$f(x_1, x_2, \cdots, x_n) \qquad \text{where } x_i \in Dom_i \qquad (6)$$

Where Dom_i represents the domain of the i-th (real valued) parameter x_i. For this type of optimization of real valued functions, a real value GA method is used (which is more efficient than the usual bit-string GA [Wright91]). We use this real valued GA approach in our integration of local search techniques.

Each bug in our BUGS2 program is characterized by 3 parameters :-

$$Bug_i(t) : \text{position } \vec{X}_i(t) = (x_1^i(t), \cdots, x_n^i(t)) \qquad (7)$$

$$\text{directional} - \text{code } \vec{DX}_i(t) = (dx_1^i(t), \cdots, dx_m^i(t)) \qquad (8)$$

$$\text{energy} \qquad e_i(t), \qquad (9)$$

where t is the (GA) generation count of the bug, x_i is its i-th position component in the search space, and $\vec{DX}_i$ is the directional control code which is used to determine the incremental changes in the bug positions. The updated position is calculated as follows :-

$$\vec{X}_i(t+1) = \vec{X}_i(t) + G(\vec{DX}_i(t)) \qquad (10)$$

Where the function G maps the directional-code $\vec{DX}$ to the direction vector. For instance, in the case of the predatory behavior described in section 2, G implements the random move and head-turning mechanisms, where $\vec{DX} = (dx_1(t), \cdots, dx_8(t))$. The fitness of each bug is derived with the aid of the function (3). The energy $e_i(t)$ of bug "i" at time (or GA generation) "t" is defined to be the cumulative sum of these function values with some aging decay (see Step3 below).

With the above definitions, the BUGS2 algorithm can now be introduced.

Step1 The initial bug population is generated with random values.
$Pop(0) = \{Bug_1(0), \cdots, Bug_{N0}(0)\}$,
where N_0 is the population size at GA generation 0.
The generation time is initialized to $t := 0$.

Step2 Move each bug using (7) synchronously.

Step3 The fitness is derived using (3) and the energy is accumulated using

$$for\ i := 1\ to\ N_t\ do\ \ e_i(t) := e_i(t-1) + f_i(\vec{X(t)}) - C_{age}, \qquad (11)$$

where C_{age} is the aging constant.

Step4 If some bug $Bug_j(N_t)$ has negative energy, it dies;
$Pop(t) := Pop(t) - \{Bug_j(N_t)\}, N_t := N_t - 1$.

Step5 If there are some reproducible bugs, call *BUGS-GA(t)*.

Step6 $Pop(t+1) := Pop(t)$, $t := t+1$, $N_{t+1} := N_t$, and then go to step2.

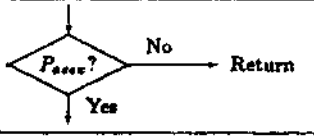
In **STEP1**, the initial bugs are generated using a uniform random generator for setting $\vec{X}$ and $\vec{DX}$. In order to reduce the preponderance of asexual reproduction,

Sexual reproduction :

	individual		energy	gene			individual		energy	gene
$Parent_1$	$Bug_i(t)$	:	E_1	G_1	⇒	$Parent'_1$	$Bug_n(t+1)$	:	$\frac{E_1}{2}$	G_1
$Parent_2$	$Bug_j(t)$	:	E_2	G_1		$Parent'_2$	$Bug_{n+1}(t+1)$	:	$\frac{E_2}{2}$	G_2
						$Child_1$	$Bug_{n+2}(t+1)$	:	$\frac{E_1+E_2}{4}$	G'_1
						$Child_2$	$Bug_{n+3}(t+1)$	:	$\frac{E_1+E_2}{4}$	G'_2

Reproduction condition ⇒ (E_1 > Reproduction energy threshold) ∧ (E_2 > Reproduction energy threshold) ∧ (distance between Parent1 and Parent2 < Reproduction Radius)

Recombination ⇒
G'_1, G'_2 : uniform crossover of G_1 and G_2 with P_{cross}, mutated with P_{mut}



Asexual reproduction :

	individual		energy	gene			individual		energy	gene
Parent	$Bug_i(t)$	:	E	G	⇒	$Child_1$	$Bug_n(t+1)$	:	$\frac{E}{2}$	G'
						$Child_2$	$Bug_{n+1}(t+1)$	:	$\frac{E}{2}$	G''

Reproduction condition ⇒ (E > producible energy)

G', G'' : mutation of G with P_{mut}

Table 2 Flow chart of BUG-GA

asexual reproduction can only occur with a probability p_{asex}

The BUGS2 version of the Genetic Algorithm *BUGS-GA(t)* is shown in Table 2.

This subroutine works in much the same way as a real valued GA, except that it operates on the directional code *(DX)*, and not on the positional vector (X). Positions are thus untouched by the adaptive process of the GA, and are changed only gradually as a result of *DX* increments. On the other hand, the fitness is evaluated using the positional potential, which is the same as for a real valued GA. Furthermore, chromosome selection is based on the cumulative fitness, i.e. the energy (see (8)). To summarize, the difference between a usual real valued GA [Wright91] and BUGS2 is as follows :-

	Fitness	*Chromosome*	*Selection*
BUGS2	$\vec{X}$	$D\vec{X}$	Energy
real GA	$\vec{X}$	$\vec{X}$	Fitness

Table 3 BUGS2 vs. real valued GA

The main difference lies in the GA chromosome (i.e. $\vec{X}$ v.s. $D\vec{X}$) and the selection criteria (i.e. energy vs. fitness). These differences will play a major role later in this paper when we attempt to combine a hill-climbing mechanism with the usual adaptive GA approach. The basic idea of this combination is derived from the previous experimental results in section 2.

4 Experiments with BUGS2

This section describes some experiments for computer vision applications. The aim of the first experiment is to derive multiple descriptors from noisy vision data (See Fig.5(a)). Multiple-line fitting problems are more difficult to solve than single line fitting problems and cannot be solved easily with usual optimization techniques such as the simulated annealing. This is because it is necessary to seek different local optima simultaneously. On the contrary, BUGS2 can solve this type of problem naturally by introducing biologically realistic features; such as resource competition and resource sharing.

The bugs used to solve multiple-line fitting problems are defined as follows:-

$$Bug_i(t) : \text{position } \vec{X}_i(t) = (r^i(t), \theta^i(t)) \quad (12)$$

$$\text{direction} - \text{code } D\vec{X}_i(t) = (dx_1^i(t), \cdots, dx_6^i(t)) \quad (13)$$

$$\text{energy } e_i(t), \quad (14)$$

Each bug position provides two parameters (r, θ) for the equation of a straight line, where r is the distance from the origin to the line, and θ is the angle of the normal to the line. With some simple trigonometry, it can be shown that the equation of the line is:-

$$(cos\theta)x + (sin\theta)y = r \quad (15)$$

Hence there is a mapping (called Hough transformation) from a bug's position (r, θ) to the equation of a line. The positions and energies of these bugs are derived as explained in the earlier bug simulations. The fitness of each "line finding" bug in (r, θ) is defined as follows:-

$$f(r, \theta) = no.\ of\ points\ on\ the\ line\ (cos\theta)x + (sin\theta)y = r \quad (16)$$

Instead of raw vision data, we use data pre-processed by smoothing techniques such as the "median filter" [Nevatia82]. The parameters for experiments are as follows. We used a selection process to weed out hopeless (i.e. low fitness) bugs (Selection Period and Selection Rate).

r (domain range)	0.0 ~ 1.0	P_{CROSS}	0.6
θ (domain range)	0.0 ~ 2π	$P_{MUTATION}$	0.033
line points	30 ~ 80	P_{asex}	0.1
random points	10 ~ 20	*Selection Period*	100
Init. pop. size	40	*Selection Rate*	0.8
Int. Energy	5	*Repr. Energy*	15
Aging decay	0.6	*Repr. Raidus*	0.1

Table 4 Parameters

Fig.5(d) shows the results for three-line fitting problems. Each small "streak" in the (r, 0)-plane represents one bug. The tail of each bug represents *DX* (length and direction) and the dot size indicates fitness (the larger fitness, the bigger dot). Bugs share resources and compete with each other appropriately, which means they hill-climb over the r — *6* plane. Fig.5(c) illustrates the approximate search space which maps the fitness *f(r,0)* (vertical axis) in the *r — 0* plane. Note that different "species" of bugs correspond to these optima.

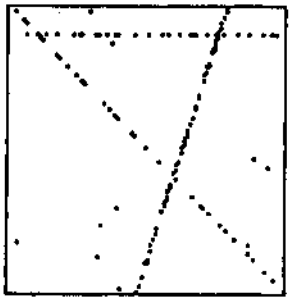
In a second experiment, we tried to derive a model description using (super)quadric forms from vision data (normals and positions). The (super)ellipsoidal shape model is represented in total by these eleven independent parameters. Fitness value is defined using the error-of-fit measure in depth and surface orientation. The results (such as shown in Fig.7) were so satisfactory that we have confirmed the effectiveness of BUGS2. See [Iba92a,92d] for more details of this experiment.

5 Discussion

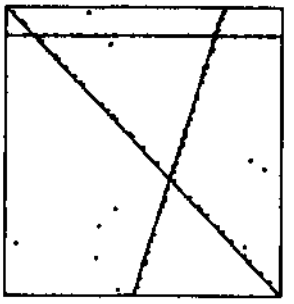
We conducted further experiments to show how the bugs in our BUGS2 program evolved, and how they were used to solve certain optimization problems [Iba92a]. To facilitate comparison with earlier GA optimization solutions, we used several "benchmark" optimization problems, such as Traveling Salesman Problem (TSP), N-queen problems, and DeJong's standard functions [Booker87]. These empirical studies highlighted the advantage of BUGS2 over usual GA approaches:-

1. Direction-based GAs establish an effective "coarse to fine" search approach. This is realized by tail-shrinking mechanisms, in which directional code vectors *DX* are adaptively changed over the generations, so as to switch from global to local search.
2. Energy-based selection integrates GA search with hill-climbing mechanisms.
3. For search in higher dimensions, using direction-based GAs (with energy-based selection) lead to adaptive acquisition of only the essential search dimensions, which are known to be relatively few in general [Rechenberg86].

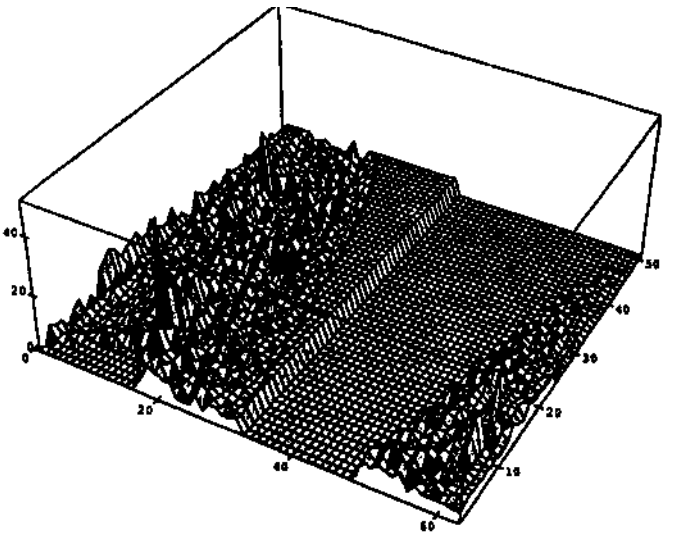
Therefore we believe that BUGS2 can be applied to wider and more practical areas. Furthermore, this direction-based evolution is somewhat similar to the



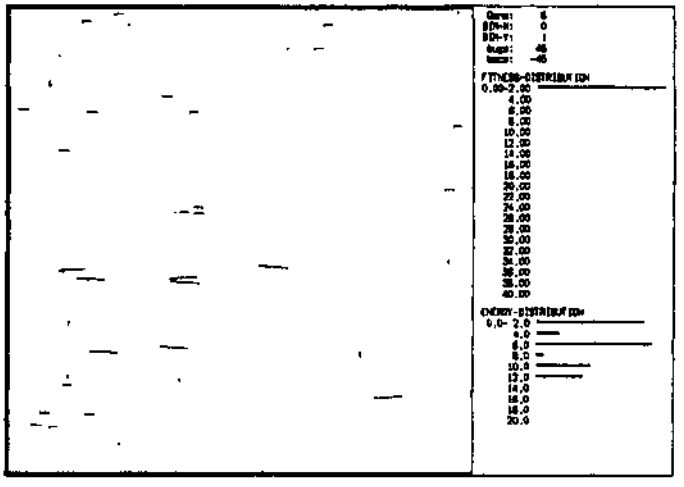
(a) Noisy data



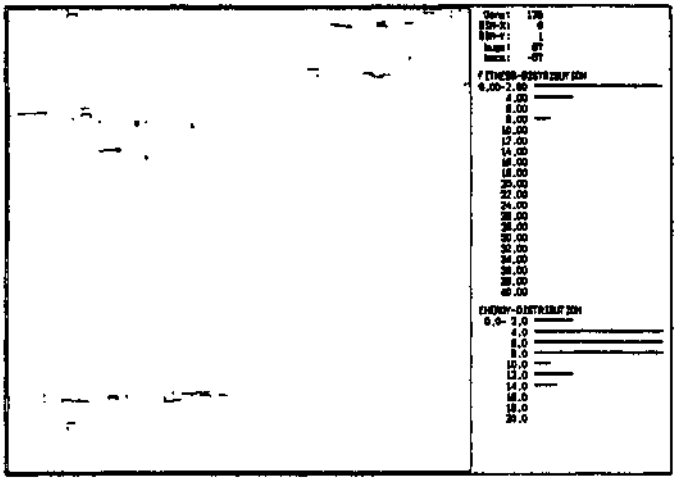
(b) Acquired lines



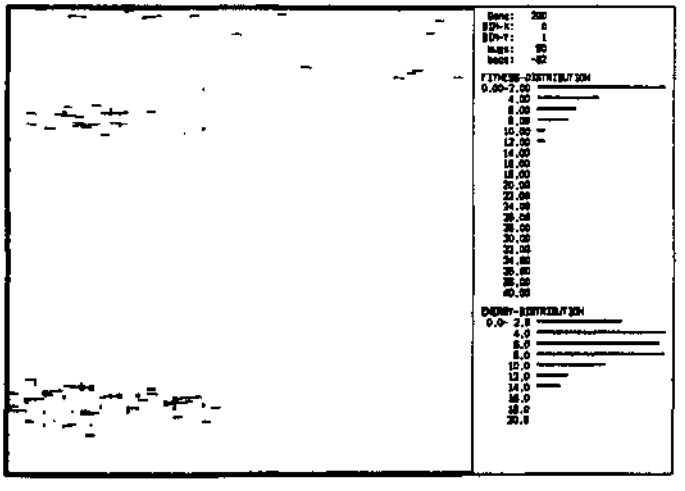
(c) Search space



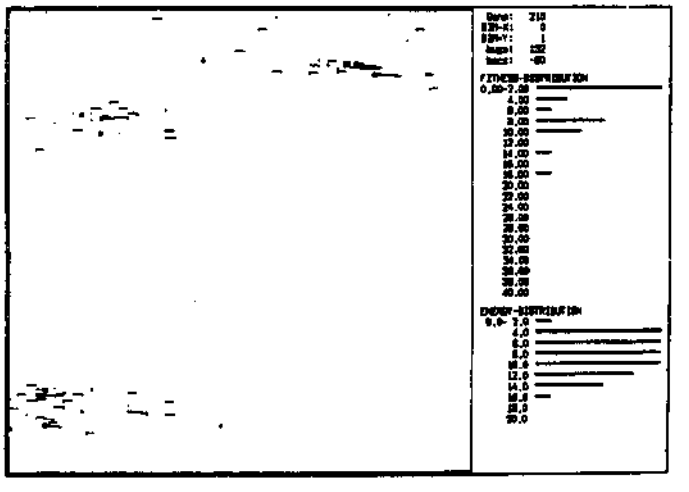
6 Generations



125 Generations



200 Generations



215 Generations

(d) Experimental results

Fig.5 Three-line fitting problem

strategic learning of a meta-GA. Informally, a meta-GA's search control can be regarded as a generalization of the adaptation of the bugs' directions. Current work involves extending the BUGS2 approach to a more structured (hierarchical) form of strategic learning, e.g using meta-GAs [Iba92b].

6 Conclusion

This paper has described a system called BUGS2 which has combined the adaptive nature of traditional GA type search, with a hill-climbing mechanism. The basic idea of this combination is derived from the simulation of bugs which learn to hunt bacteria. We experimented with a "bug searcher" program as a preliminary study, and found that bugs evolve different types of search strategies for different niches. In order to show how well BUGS2 performs, we undertook several experiments, e.g. line fitting, and the recovery of shapes in a computer vision application. To improve the evolution of the behavior of the bugs, we introduced some ideas for making the bugs more biologically realistic.

We believe that our BUGS2 system can be applied to broader and more practical areas, such as symbolic learning or the acquisition of meta-strategies. Further research on these ideas is currently under way.

Acknowledgment.
We have profit ted from many suggestions and discussions with members of our section and laboratory. We also would like to thank Thomas Weigert for helping us to improve the English of this paper, Sumitaka Akiba for helping us with the graphical implementation of BUGS2 and Yumiko Owada for rearranging figures.

References

|Ackley87] Ackley, D. H. A connectionist machine for genetic hillclimbing, Kluwer Academic Publishers, 1987

[Booker87] Booker, L. Improving search in genetic algorithms, In Genetic Algorithms and Simulated Annealing, (ed. Davis, L.), pp.61-73, Morgan Kaufmann, 1987

[Caruana et al.88] Caruana, R.A. and Schaffer, J.D. Representation and hidden bias: gray vs. binary coding for genetic algorithms, In Proc. of 5th conference on machine learning, 1988

[Dewdney89] Dewdney, A. K. Computer recreations, simulated evolution: wherein bugs learn to hunt bacteria, In Scientific american, pp. 104-107, 1989 May

[Goldberg89] Goldberg, D. E. Genetic algorithms in search, optimization and machine learning, Addison Wesley, 1989

[Iba et a/.91] Iba, H. and Inoue, H. Reasoning of geometric concepts based on algebraic constraint-directed method In Proc. of 12th IJCAI, pp.143-149, Morgan Kaufmann, 1991

[Iba et a/.92a] Iba, H. Akiba, S. Higuchi, T. and Sato, T. BUGS : A bug-based search strategy using genetic algorithms, ETL-TR92-8, in Parallel Problem Solving from Nature, North-Holland, 1992

[Iba et a/.92b] Iba, H. and Sato, T. Meta-level strategy learning for GA based on structured representation, ETL-TR92-12, in Proc. of Pacific Rim International Conference on Artificial Intelligence (PRICAI92), 1992

[Iba et a/.92c] Iba, H. deGaris, and H. Higuchi, T. Evolutionary learning of predatory behaviors based on structured classifiers, in Proc. of Second International Conference on Simulation of Adaptive Behavior, MIT Press, 1992

[Iba et al.92d] Iba, H. deGaris, H. Higuchi, T. and Sato, T. Evolutionary learning strategy using bug-based search, ETL-TR92-24, 1992

[Mtihlenbein89] Miihlenbein, H. Parallel genetic algorithms, population genetics and combinatorial optimization, In Proc. of 3rd /CGA,Morgan Kaufmann, 1989

[Nevatia82] Nevatia,R. Machine perception, Prentice-Hall, 1982

[Rechenberg86] Rechenberg, I. Evolution strategy and human decision making, In Human decision making and manual control, (ed. Willumeit, H. P.), North-Holland, 1986

[Shaffer et al.9l] Schaffer, J.D. Eshelman, L.J. and Offutt, D. Spurious correlations and premature convergence in genetic algorithms, In Foundations of genetic algorithms, (ed. Rawlins J.E.), Morgan Kaufmann, 1991

[Wright9l] Wright, A.H. Genetic algorithms for real parameter optimization, In Foundations of genetic algorithms, (ed. Rawlins J.E.), Morgan Kaufmann, 1991

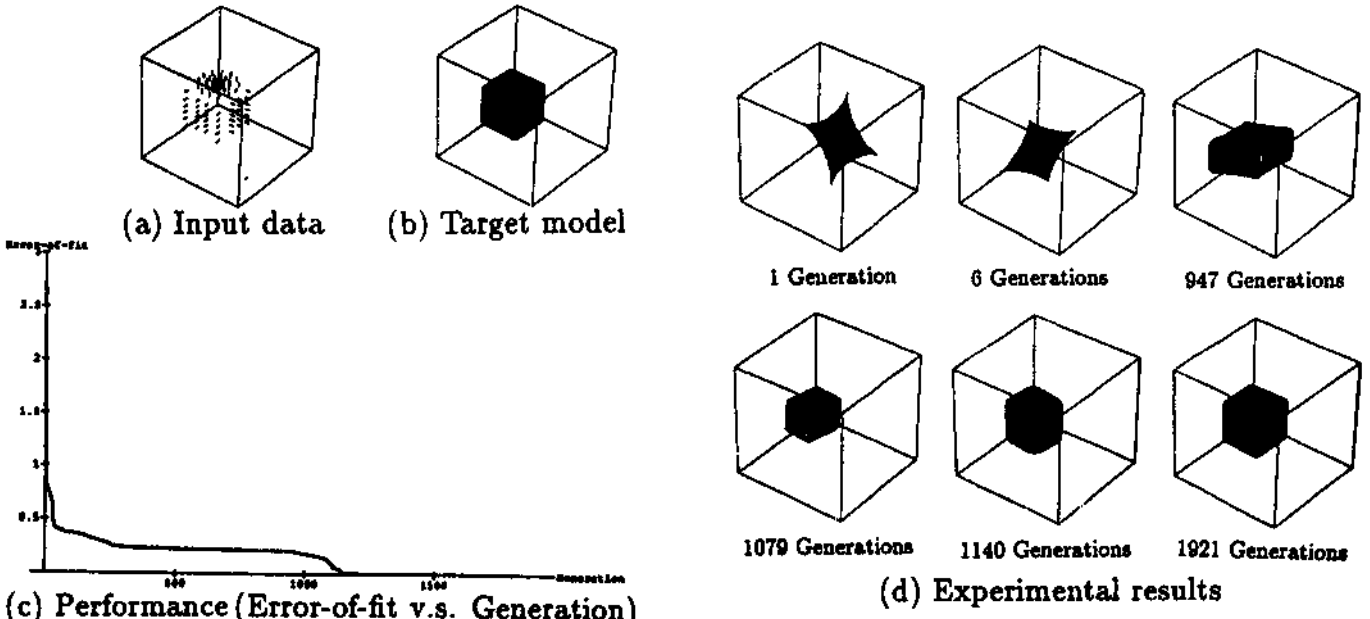


Fig.6 Recovery of cubic model